John Broxton

PDX Code Guild: Capstone Project Proposal

Tessera

updated October 20, 2015

Elevator Pitch

The Image Processor is an easy-to-use web app that accepts and returns images in an assembled format that could be used to create banner- or billboard-sized prints. This app has two main goals: 1.) to make it easier for users to interface with a large number of images and 2.) use technology to reduce the time and/ or cost of creating a large-scale print from the assemblage of these images. (A large-scale print is defined here as anything larger than 5'x5'.)

Overview

When people think about photography, generally the first thing that comes to mind is the image they can capture with their phone. Pretty soon over a trillion photographs will be taken with phones. What is less well known is the actual print size that can be covered by the billions of photos that are taken every year. Consider that a square image taken with an iPhone 4s is 2500 x 2500 pixels (roughly 7 megapixels). An image of this size could be printed at 20" x 20" without an appreciable drop in quality. the image would be about 12" x 12". So, 30 photographs could cover an 8' x 10' wall.

In 2015 over one trillion photographs will be captured with smartphones. This means that every day millions of great digital images will be forgotten in the cloud or on a device because dealing with multiple images is a time-intensive and laborious process.

The Image Processor aims to provide tools for making the editing of multiple photos as easy as editing a single photo. Currently, the only way to create an assembly or collage of photos is to pay a designer a considerable sum or do all the resizing, cropping, and image manipulation yourself in a photo editor. The options in the web app will be minimal, with most of the

processor-intensive steps happening on the back end. The program will resize images based on the number uploaded. The user will have an option to specify the size of their assembled image. In contrast to the user's photo stream or folder, the front-end interface will be clean and uncluttered.

Use Case

You're a motivated administrator working at a company with a strong culture of social cohesion and employee empowerment. While helping to plan the company retreat, you ask a few of your more visually-oriented coworkers to take pictures and store them in a cloud folder. You end up with 100 images of varying quality. After creating an account on the Image Processor, you upload the photos and after several moments, you're rewarded with a mapping interface that shows your photos assembled in a large grid. The controls are familiar, pinch or use the scroll wheel to zoom in and out, swipe or click to orient on the screen. You're not satisfied with the arrangement of the images, so you investigate the controls. There are three configurations of images, and three options for image manipulation. You select the square-cropped grid in black and white. It's much improved; for the final touch you move a couple of the group shots to the center. Perfect. You select 8' X 10', pay a nominal fee, and submit your order. From the upload to the submission, you've spent 10 minutes.

Within a week you receive a shipping tube containing three 8' X 42" sheets of your image assembly printed on Phototex self-adhesive fabric material. The adhesive is semi-permanent, so it can be installed and removed dozens of times and leaves no residue. You can't decide if it should be placed in a high-traffic location to give potential clients a window into the company's culture, or in the breakroom to boost company morale. You end up placing another order after the enthusiastic response.

Goals

1. **Provide online tools to cut the time and energy it takes to create a large-scale photo assembly:** The Image Processor will give users the ability to deal with a plethora of images that would otherwise be considered unmanageable.
2. **Return the photo assembly as quickly as possible:** Logic should be clean and efficient to reduce wait times.
3. **Offer solutions for creating physical copies of photo assemblies:** Even the biggest computer monitor doesn't convey the power of a large-scale image creation - you need a physical print to do that.

Specifications

The Image Processor will use the Python Imaging Library framework, a.k.a., Pillow.

https://pillow.readthedocs.org/

Contributing to image manipulation will be PythonMagick, an object-oriented Python interface to ImageMagick, an open-source software suite. Specifically, the *montage, composite, transform,* and *special effects* programs in IM.

http://www.imagemagick.org/.

The assembled image will be viewable via Google Map ImageCutter:

http://www.e-lucid.com/i/software/image_publishing/gmap_image_cutter.html

An example of ImageCutter in action is available here:

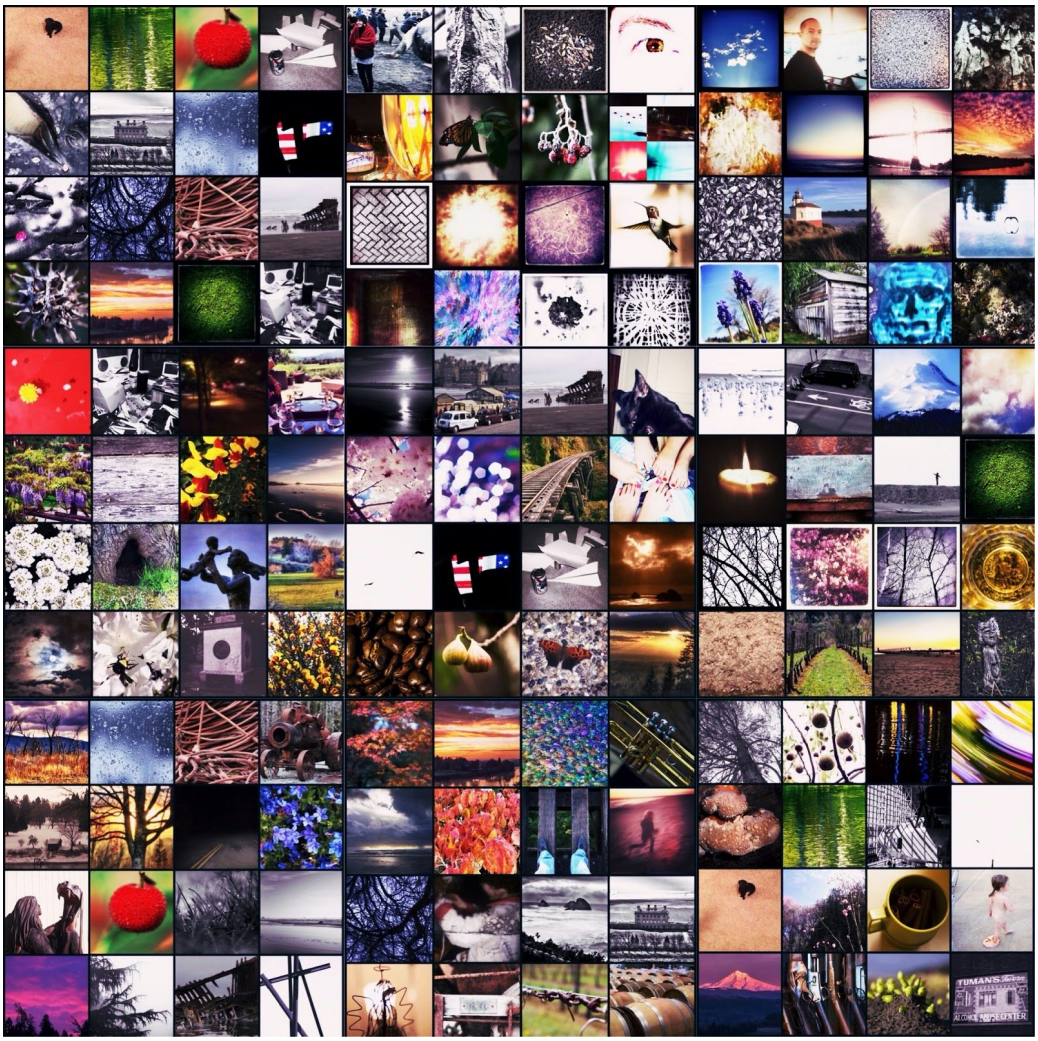
http://www.casa.ucl.ac.uk/googlemaps/house.html

The first option upon launch will allow the user to order their photo montage printed on Phototex self-adhesive fabric material:

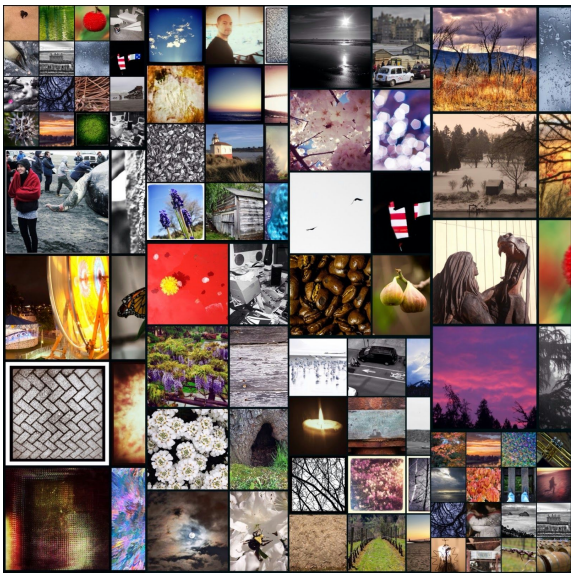
http://www.phototexgroup.com/

Examples

A montage of 144 images arranged in a square.



Example images rendered with a black & white filter, and with images of varying sizes



Milestones

1. 5 weeks

   The web app will accept photos in PNG, JPEG, GIF, TIFF, and SVG formats and return a large-scale composite image, with an option to order the returned image as a banner printed on Phototex fabric material. The photos will be placed on a grid, either cropped into squares, or in their original aspect ratio with a border.

2. 6 months

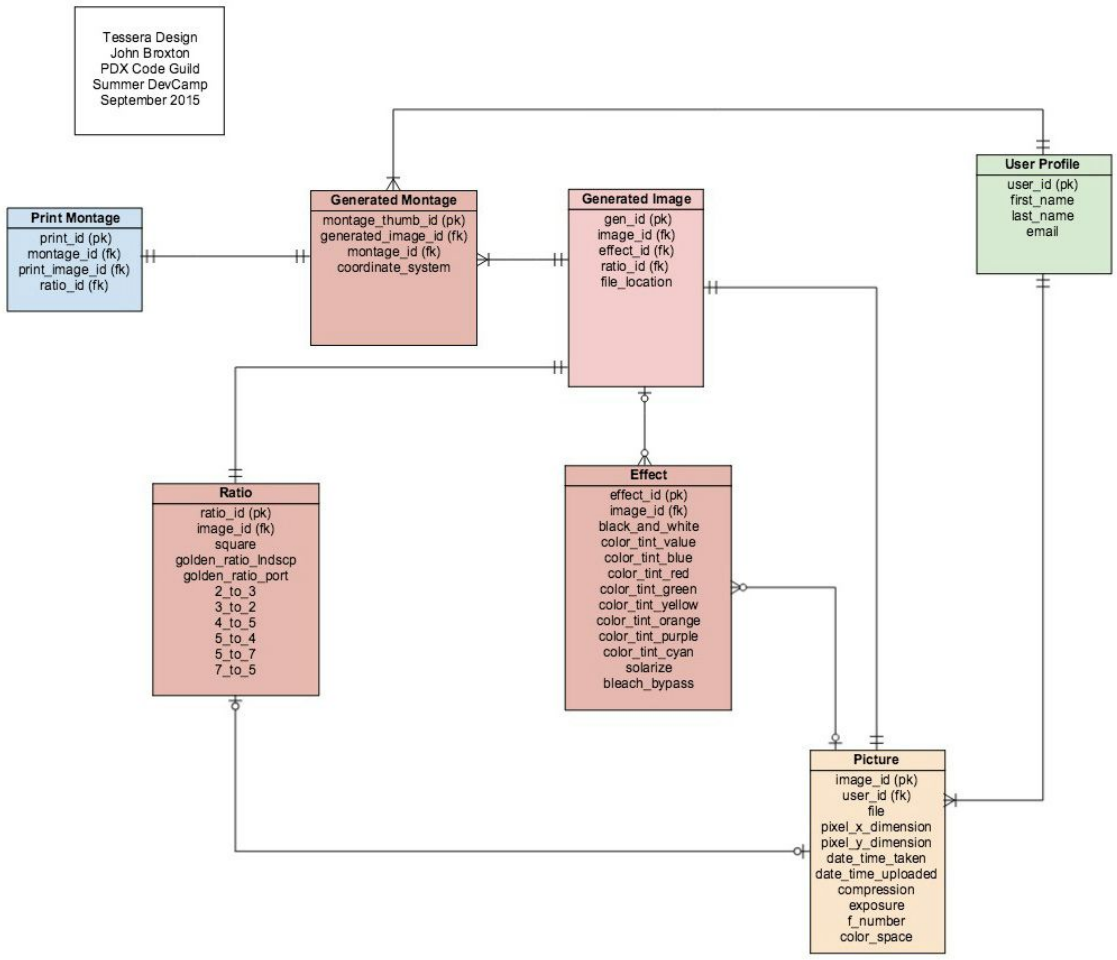
   Create logic that will randomize the placement of photos based on exif data. For example, photos that were taken at the same time will be grouped together. The logic will create nodes on the canvas around which the photos are placed at random lengths, as well as transparency on certain photos that are ‘ghosted’ into the

background to give the canvas depth. In addition, create ‘kaleidoscope’ logic that will divide photos into triangular patterns.

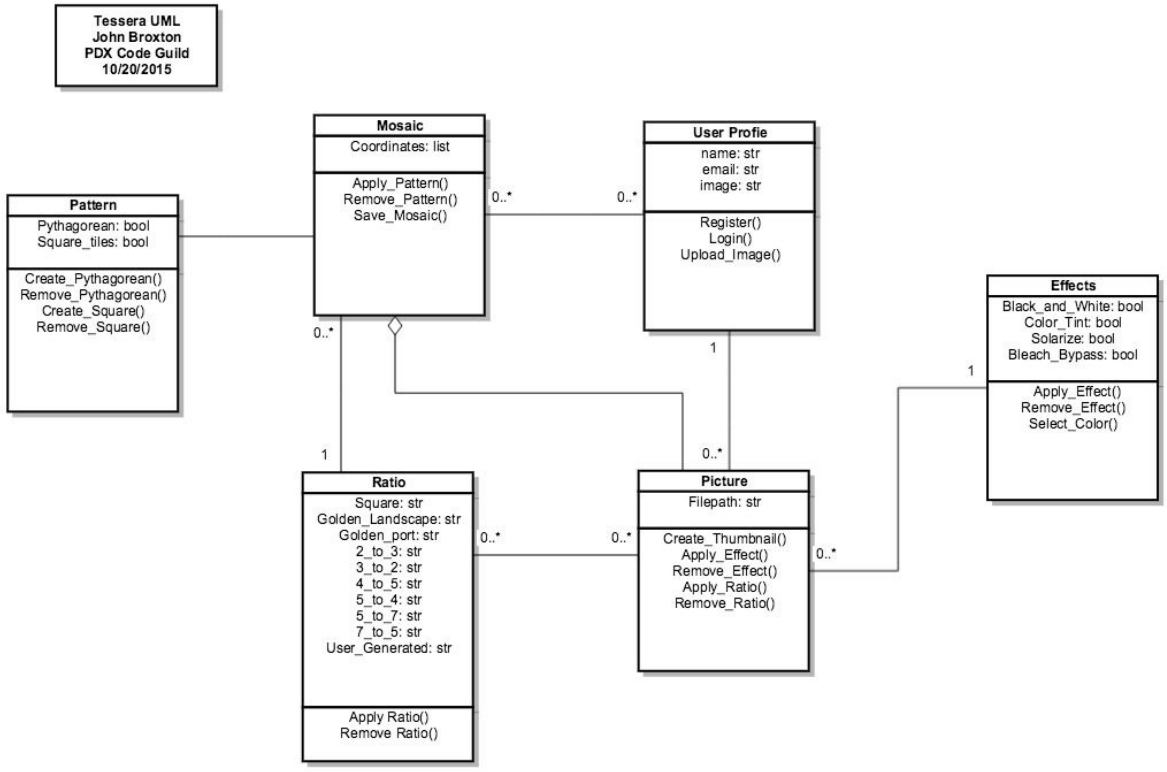
3. 10 months

   Create standalone app that will access camera functionality to process photos upon taking them, eliminating the need to upload the photos via webapp.

Database Design



UML Class Diagram


Tessera UML
John Broxton
PDX Code Guild
10/20/2015
Mosaic
Coordinates: list
Apply_Pattern()
Remove_Pattern()
Save_Mosaic()
User Profie
name: str
email: str
image: str
Register()
Login()
Upload_Image()
Pattern
Pythagorean: bool
Square_tiles: bool
Create_Pythagorean()
Remove_Pythagorean()
Create_Square()
Remove_Square()
Effects
Black_and_White: bool
Color_Tint: bool
Solarize: bool
Bleach_Bypass: bool
Apply_Effect()
Remove_Effect()
Select_Color()
Ratio
Square: str
Golden_Landscape: str
Golden_port: str
2_to_3: str
3_to_2: str
4_to_5: str
5_to_4: str
5_to_7: str
7_to_5: str
User_Generated: str
Apply Ratio()
Remove Ratio()
Picture
Filepath: str
Create_Thumbnail()
Apply_Effect()
Remove_Effect()
Apply_Ratio()
Remove_Ratio()
0..*
1